

# INFORMATICA

# Clase de hoy

- Repaso de reserva de memoria (estática y dinámica)
- Resolución práctica números primos
- Operaciones con vectores y matrices
- Práctica: multiplicación de una matriz por un vector
- Jueves 26-10-17 -- Examen:
  - Aula E1
  - Horario de clase
  - Entrega de código al final del examen a través de Moodle
  - El examen se realizará en vuestro portátil
    - (BATERÍA – INTERNET)

# Arrays: Vectores y Matrices: Declaración II

- Parámetro

```
program declaracion

integer, parameter :: n = 2
integer, parameter :: m = 3
integer, parameter :: Linf = 0
integer, parameter :: Lsup = 10

integer :: U(n)
real :: V(n*m)
real :: A(n,m)
real :: T(Linf:Lsup, Linf:Lsup)

end program declaracion
```

- Dinámica (luego con más detalle)

```
program declaracion

integer :: n
Integer :: m

integer, allocatable :: U(:)
real,    allocatable :: A(:, :)

end program declaracion
```



# Arrays: Vectores y Matrices: Declaración II

- Parámetro

```
program declaracion

integer, parameter :: n = 2
integer, parameter :: m = 3
integer, parameter :: Linf = 0
integer, parameter :: Lsup = 10

integer :: U(n)
real :: V(n*m)
real :: A(n,m)
real :: T(Linf:Lsup, Linf:Lsup)

end program declaracion
```

- Dinámica (luego con más detalle)

```
program declaracion

integer :: n
Integer :: m

integer, allocatable :: U(:)
real, allocatable :: A(:, :)

end program declaracion
```



$U(:)$  es un vector y  $A(:, :)$  una matriz de tamaño desconocidos al principio del programa

# Arrays: Vectores y Matrices: Asignación II

- Parámetro

```
program declaracion
```

```
integer, parameter :: n = 2  
integer, parameter :: m = 3
```

```
integer :: U(n)  
real :: A(n,m)
```

```
A = 0.d0  
U = 0.d0
```

- Igual que antes

```
end program declaracion
```

- Dinámica (luego con más detalle)

```
program declaracion
```

```
integer :: n  
Integer :: m
```

```
integer, allocatable :: U(:)  
real, allocatable :: A(:, :)
```

```
! Cuerpo de programa
```

```
n = 2
```

```
m = 3
```

```
allocate(U(n))
```

```
allocate(A(n,m))
```

```
A = 0.d0
```

```
U = 0.d0
```

```
end program declaracion
```



# Arrays: Vectores y Matrices: Asignación II

- Parámetro

```
program declaracion
```

```
integer, parameter :: n = 2  
integer, parameter :: m = 3
```

```
integer :: U(n)  
real :: A(n,m)
```

```
A = 0.d0  
U = 0.d0
```

- Igual que antes

```
end program declaracion
```



- Dinámica (luego con más detalle)

```
program declaracion
```

```
integer :: n  
Integer :: m
```

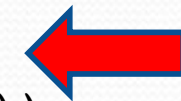
```
integer, allocatable :: U(:)  
real, allocatable :: A(:, :)
```

```
! Cuerpo de programa
```

```
n = 2
```

```
m = 3
```

```
allocate(U(n))  
allocate(A(n,m))
```



Reserva dinámica  
de memoria para  
U(:) y A(:, :)

```
A = 0.d0
```

```
U = 0.d0
```

```
end program declaracion
```

# Arrays: Vectores y Matrices: Asignación III

- Asignación por bucle

```
program asignacion

integer, parameter :: n = 10
integer :: U(n)
integer :: i

!--- Fin declaracion -----

do i=1,n
    U(i) = i*i
end do

write(*,*) U

do i=1,n
    write(*,*) U(i)
end do

end program asignacion
```



# Arrays: Vectores y Matrices: Asignación III

Escribir un programa que defina la matriz  $A_{n \times n}$  con  $n = 100$

$$a_{ij} = \begin{cases} i + j & \text{Si } i \leq j \\ 0 & \text{Si } i > j \end{cases}$$

$$A = \begin{pmatrix} 2 & 3 & 4 & \cdots & 101 \\ 0 & 4 & 5 & \cdots & 102 \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 200 \end{pmatrix}$$



# Arrays: Vectores y Matrices: Asignación III

Escribir un programa que defina la matriz  $A_{n \times n}$  con  $n = 100$

$$a_{ij} = \begin{cases} i + j & \text{Si } i \leq j \\ 0 & \text{Si } i > j \end{cases} \quad A = \begin{pmatrix} 2 & 3 & 4 & \cdots & 101 \\ 0 & 4 & 5 & \cdots & 102 \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 200 \end{pmatrix}$$

```
program main
integer, parameter :: n = 100
integer :: A(n,n)
integer :: i, j
!--- Fin declaracion -----

do i=1,n
    do j=1,n
        if (i <= j) then
            A(i,j) = i + j
        else
            A(i,j) = 0
        end if
    end do
end do
end program main
```

# Arrays: Vectores y Matrices: Asignación III

Escribir un programa que defina la matriz  $A_{n \times n}$  con  $n = 100$

$$a_{ij} = \begin{cases} i + j & \text{Si } i \leq j \\ 0 & \text{Si } i > j \end{cases} \quad A = \begin{pmatrix} 2 & 3 & 4 & \cdots & 101 \\ 0 & 4 & 5 & \cdots & 102 \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 200 \end{pmatrix}$$

```
program main
integer, parameter :: n = 100
integer :: A(n,n)
integer :: i, j
!--- Fin declaracion -----

A = 0
do i=1,n
    do j=1,n
        if (i <= j) A(i,j) = i + j
    end do
end do
end program main
```



# Arrays: Vectores y Matrices: Asignación III

Escribir un programa que defina la matriz  $A_{n \times n}$  con  $n = 100$

$$a_{ij} = \begin{cases} i + j & \text{Si } i \leq j \\ 0 & \text{Si } i > j \end{cases} \quad A = \begin{pmatrix} 2 & 3 & 4 & \cdots & 101 \\ 0 & 4 & 5 & \cdots & 102 \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 200 \end{pmatrix}$$

```
program main
integer, parameter :: n = 100
integer :: A(n,n)
integer :: i, j
!--- Fin declaracion -----

A = 0
do i=1,n
    do j=i,n
        A(i,j) = i + j
    end do
end do
end program main
```

# Asignación Dinámica

```
program declaracion
```

```
integer :: n
```

```
Integer :: m
```

```
integer, allocatable :: U(:)
```

```
real, allocatable :: A(:, :)
```

```
! Cuerpo de programa
```

```
allocate(U(n))
```

```
allocate(A(n,m))
```

```
A = 0.d0
```

```
U = 0.d0
```

```
end program declaracion
```

● Inconvenientes de la declaración de un *array* con tamaño fijo:

● Si el tamaño prefijado es mayor que el número de valores que se van a almacenar, estamos *malgastando* memoria.

● Si el tamaño prefijado es menor que el número de valores que se van a utilizar, el programa dará un error de ejecución.

● Solución en Fortran: ***arrays dinámicos***.



# Asignación Dinámica

```
program declaracion
```

```
integer :: n,info
```

```
integer, allocatable :: U(:)
```

```
! Cuerpo de programa
```



Declaración de U como array dinámico

```
n = 5
```

```
allocate(U(n), stat=info)
```



Reserva de memoria, con control de error  
Si la asignación es correcta stat = 0.  
En caso contrario stat > 0

```
if (info > 0) stop '** No hay memoria &  
                & suficiente para U**'
```

```
U = 0.d0
```

```
deallocate(U, stat=info)
```



Libera la memoria previamente reservada,  
con control de error

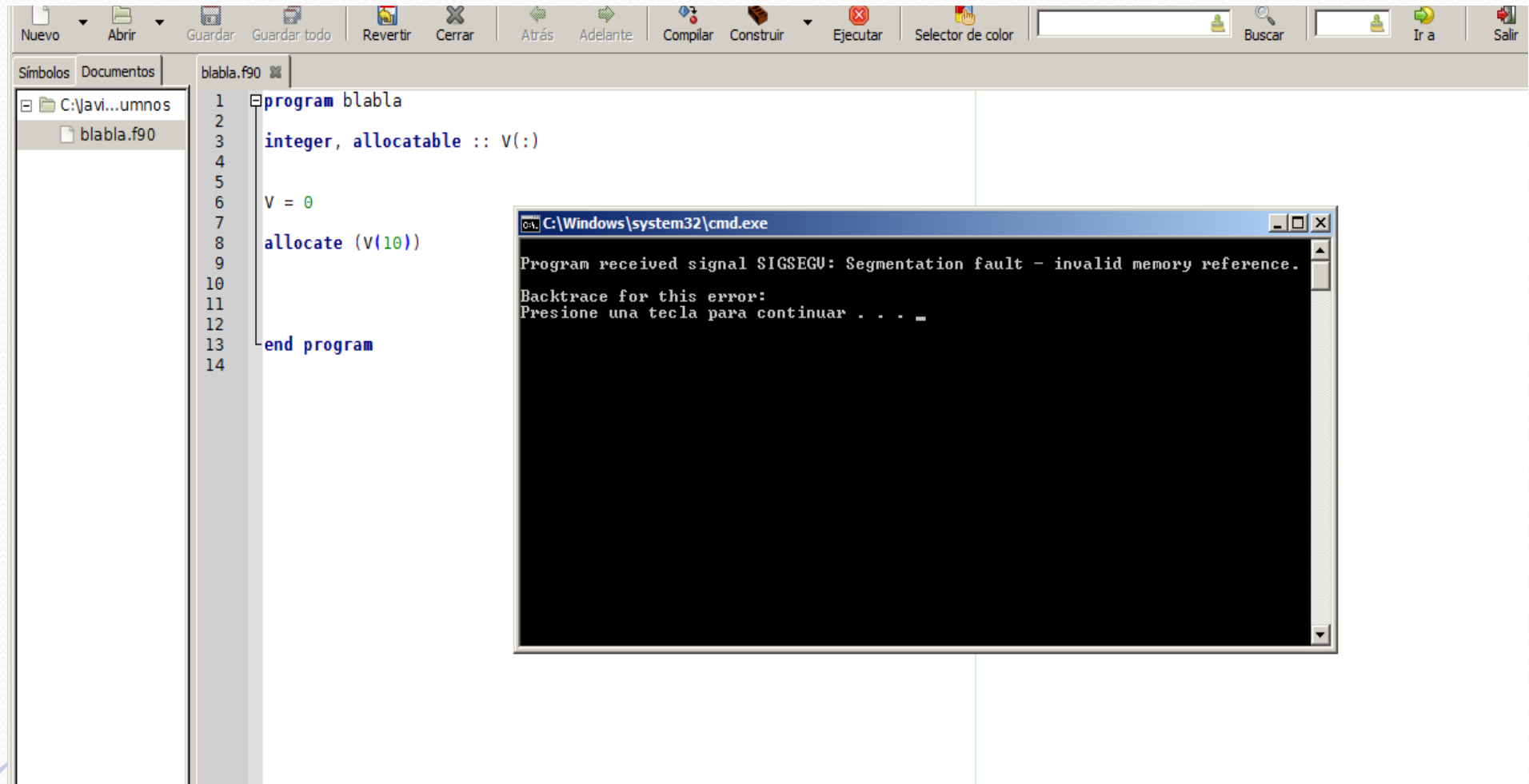
```
if (info > 0) stop '** U no tenía &  
                & memoria reservada**'
```

```
end program declaracion
```

# Estilo de programación:

## Leyendo los mensajes de error

### Reservar espacio para variables antes de usarlas



The screenshot shows a Fortran IDE with a file named `blabla.f90`. The code in the editor is as follows:

```
1 program blabla
2
3 integer, allocatable :: v(:)
4
5
6 v = 0
7
8 allocate (v(10))
9
10
11
12
13 end program
14
```

Overlaid on the IDE is a Windows command prompt window titled `C:\Windows\system32\cmd.exe`. It displays the following error message:

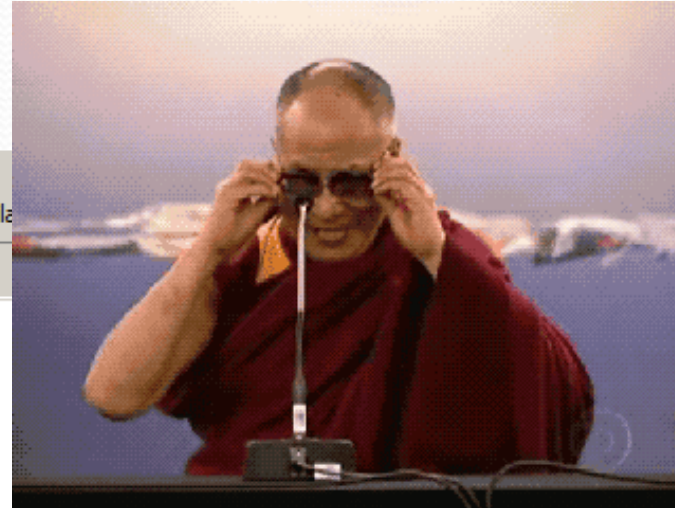
```
Program received signal SIGSEGV: Segmentation fault - invalid memory reference.
Backtrace for this error:
Presione una tecla para continuar . . . _
```



# Estilo de programación:

## Leyendo los mensajes de error

### Respetad las dimensiones.



```
blabla.f90
1  program blabla
2
3  integer, allocatable :: v(:)
4  integer :: n,m,i
5
6  n = 5
7  m = 10
8  allocate (v(n))
9
10 do i = 1, m
11
12     v(i) = i
13
14 enddo
15
16 write(*,*) v
17
18 end program
19
```

```
C:\Windows\system32\cmd.exe
At line 12 of file blabla.f90
Fortran runtime error: Index '6' of dimension 1 of array 'v' above upper bound o
f 5
Presione una tecla para continuar . . .
```

# Operaciones con vectores y matrices

```
program vectores

integer, allocatable :: u(:),v(:)
integer :: i

allocate (u(5),v(5))

do i = 1, 5
    u(i)= i
enddo
v = 1

u = u+v

write(*,*) u

end program
```



# Operaciones con vectores y matrices

```
program vectores
```

```
integer, allocatable :: u(:),v(:)  
integer :: i
```

```
allocate (u(5),v(5))
```

```
do i = 1, 5  
    u(i)= i
```

```
enddo
```

```
v = 1
```

```
u = u+v
```

```
write(*,*) u
```

```
end program
```

2 3 4 5 6

# Operaciones con vectores y matrices

```
program vectores
```

```
integer, allocatable :: u(:),v(:)  
integer :: i
```

```
allocate (u(5),v(3))
```

```
do i = 1, 5  
    u(i)= i
```

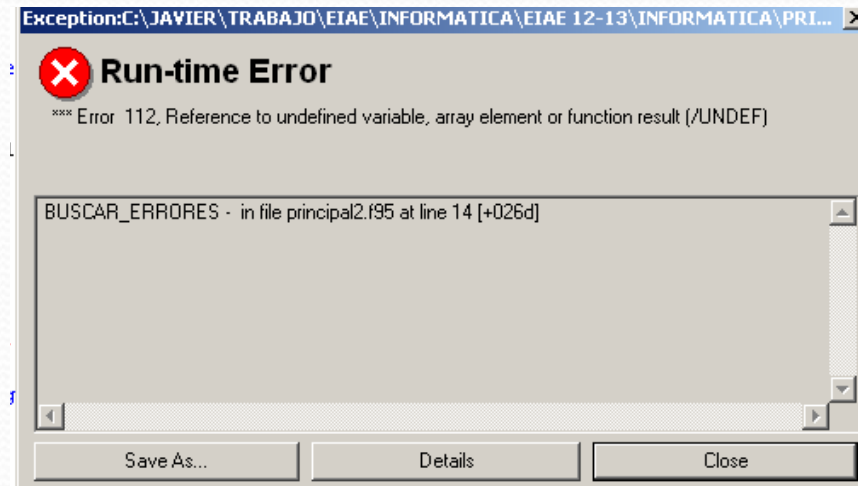
```
enddo
```

```
v = 1
```

```
u = u+v
```

```
write(*,*) u
```

```
end program
```





# Operaciones con vectores y matrices

```
program vectores
```

```
integer, allocatable :: u(:)  
integer :: i,k
```

```
allocate (u(5))
```

```
do i = 1, 5  
    u(i)= i
```

```
enddo
```

```
k = 2
```

```
u = k*u
```

```
write(*,*) u
```

```
end program
```

2 4 6 8 10

# Operaciones con vectores y matrices

```
program suma_matrices_I

integer, allocatable      :: A(:, :), B(:, :), C(:, :)
integer :: i, j

allocate (A(3,5), B(3,5), C(3,5))

A = 0
B = 0
C = 0
do i = 1, 3
    A(i,i)= 1
    do j = 1, 5
        B(i,j)=i+j
    enddo
enddo

C = A + B
do i = 1,3
    write(*,*) C(i,:)
enddo

end program
```



3	3	4	5	6
3	5	5	6	7
4	5	7	7	8



# Operaciones con vectores y matrices

```
program suma_matrices_I

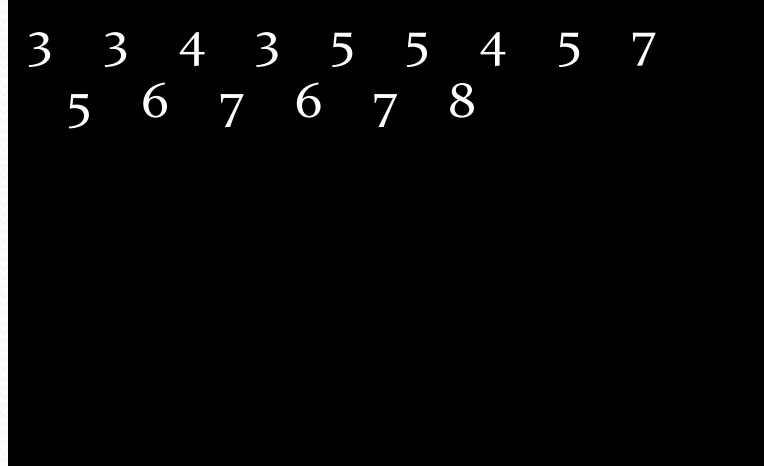
integer, allocatable      :: A(:, :), B(:, :)
integer :: i, j

allocate (A(3,5), B(3,5))

A = 0
B = 0
do i = 1, 3
    A(i,i)= 1
    do j = 1, 5
        B(i,j)=i+j
    enddo
enddo

write(*,*) A + B

end program
```



3	3	4	3	5	5	4	5	7
5	6	7	6	7	8			

# Operaciones con vectores y matrices

```
program suma_matrices_I

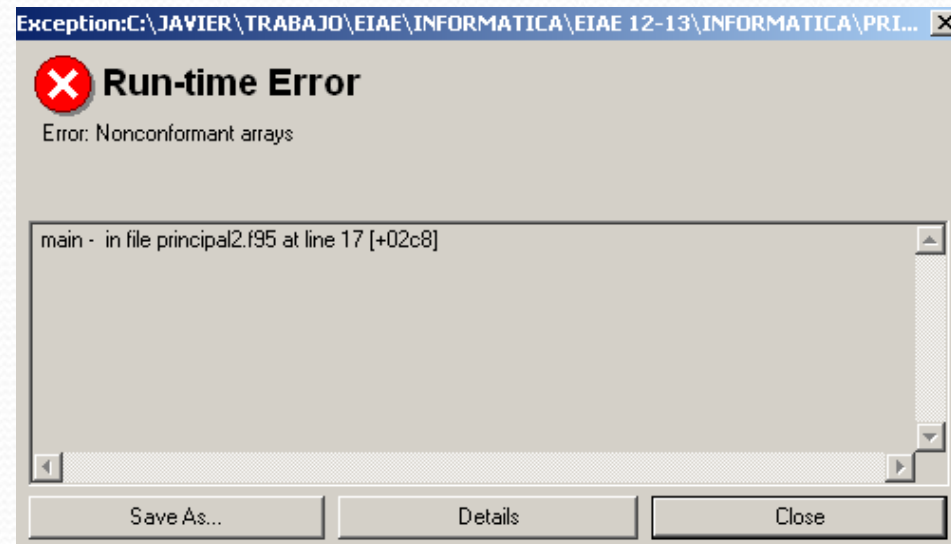
integer, allocatable :: A(:, :), B(:, :), C(:, :)
integer :: i, j

allocate (A(5, 3), B(3, 5), C(5, 5))

A = 0
B = 0
do i = 1, 3
    A(i, i) = 1
    do j = 1, 5
        B(i, j) = i + j
    enddo
enddo

C = A + B

end program
```





# Operaciones con vectores y matrices

```
program producto_matriz

integer, allocatable      :: A(:, :)
integer :: i, k

allocate (A(3,5))

A = 0
k = 7
do i = 1, 3
    A(i,i)= 1
enddo

A = k*A

do i = 1,3
    write(*,*) A(i,:)
enddo

end program
```



7	0	0	0	0
0	7	0	0	0
0	0	7	0	0

# Funciones intrínsecas.

- Ejercicio 1. crear un programa que:
  - Defina tres variables A(3,3), X(3) y b(3) que sean una matriz y un vector de reales respectivamente.
  - La reserva de memoria se realizará de forma estática.
  - Asigne los valores de A y X:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$X = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

- El programa debe calcular el producto de la matriz A por el vector X y almacenarlo en un vector b.



# Funciones intrínsecas.

- Ejercicio 2. crear un programa que:
  - Defina tres variables  $A(N,N)$ ,  $X(N)$  y  $b(N)$  que sean una matriz y un vector de enteros respectivamente.
  - La reserva de memoria se realizará de forma dinámica ( $N=100$ ).
  - Asigne los valores de  $A$  y  $X$ :

$$a_{ij} = \begin{cases} i + j & \text{Si } i \leq j \\ 0 & \text{Si } i > j \end{cases} \quad A = \begin{pmatrix} 2 & 3 & 4 & \cdots & 101 \\ 0 & 4 & 5 & \cdots & 102 \\ & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 200 \end{pmatrix} \quad X = i^2$$

- El programa debe calcular el producto de la matriz  $A$  por el vector  $X$  y almacenarlo en un vector  $b$ .